Read Me First

IMPORTANT: To install MATLAB products, you must have your Personal License Password (PLP) or License File. See page 1-2 for details.

If you are ***upgrading*** from an earlier MATLAB version:

1. Perform the installation as described in the *MATLAB Installation Guide*. We strongly recommend that you update *all* your licensed products at the same time to ensure that you have a matched set of products.
2. You can install your copy of MATLAB® 5.2 into the same directory where MATLAB 5.1 or 5.0 is installed. Do *not* install MATLAB 5.2 over a previously installed version of MATLAB 4.2.
3. Read *MATLAB 5.2 Product Family New Features* for a description of new features in 5.2; use the online Help Desk for the latest documentation.
4. Refer to this document for additional information and a description of known software problems.

If you are ***installing MATLAB for the first time***:

1. Perform the installation as described in the *MATLAB Installation Guide*.
2. Read *Getting Started with MATLAB*. Consult the rest of the printed documentation set and online Help Desk as necessary.
3. Refer to this document for additional release notes and a description of known software problems.

Computation

Visualization

Programming

Late-Breaking News for the MATLAB® 5.2 Product Family

Version 5.2

How to Contact The MathWorks:

508-647-7000	Phone
508-647-7001	Fax
The MathWorks, Inc. 24 Prime Park Way Natick, MA 01760-1500	Mail
`http://www.mathworks.com`	Web
`ftp.mathworks.com`	Anonymous FTP server
`comp.soft-sys.matlab`	Newsgroup
`support@mathworks.com`	Technical support
`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`subscribe@mathworks.com`	Subscribing user registration
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

Late-Breaking News for the MATLAB 5.2 Product Family

Printing History: January 1998 New for MATLAB 5.2

MATLAB Software Acknowledgments

MATLAB and its associated products incorporate the following third-party software:

The Delaunay function is based on code from

Steve J. Fortune (1987) A Sweepline Algorithm for Voronoi Diagrams, *Algorithmica 2*, 153-174.

The HDF capability in the functions IMREAD, IMWRITE, IMFINFO, and HDF is based on code, of which portions were developed at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign.

The JPEG capability in the functions IMREAD, IMWRITE, IMFINFO, and PRINT:

This software is based in part on the work of the Independent JPEG Group.

The TIFF capability in the functions IMREAD, IMWRITE, IMFINFO, and PRINT:

Contents

1 MATLAB 5.2

2 MATLAB Compiler 1.2

3 Simulink 2.2

4 Real-Time Workshop 2.2

5 Stateflow 1.0.6

6 Toolboxes and Blocksets

Introduction

This *Late-Breaking News* document includes information that was not available at the time the rest of the documentation set was released. This document also describes known software problems in the current release of the MATLAB 5.2 Product Family and provides workarounds whenever possible. The information in this document takes precedence over any information in the rest of the printed or online documentation.

Contents

Late-Breaking News is organized into the following six chapters:

- MATLAB 5.2
- MATLAB Compiler 1.2
- Simulink 2.2
- Real-Time Workshop 2.2
- Stateflow 1.0.6
- Toolboxes and Blocksets

MATLAB 5.2

Installation

Where to Install MATLAB 5.2

You can install your copy of MATLAB® 5.2 into the same directory where MATLAB 5.1 or 5.0 is installed.

Important: Do *not* install MATLAB 5.2 over a previously installed version of MATLAB 4.2.

Troubleshooting

If you encounter any problems during installation, visit `www.mathworks.com` to access the Technical Support Solution Engine.

Supported Platforms

Details about the supported platforms and system requirements for MATLAB 5.2 are available online via the Technical Support FAQs. You can access this information via the MATLAB 5 FAQ link from the The MathWorks home page (`www.mathworks.com`) or via the Help Desk (in The MathWorks Web Site section, via the Questions link).

Same Approach As for MATLAB 5.1

The installation process for MATLAB 5.2 is almost the same as for MATLAB 5.1. There is now an option to install a Japanese version of the documentation.

Refer to the appropriate MATLAB 5.1 *Installation Guide* for installation instructions.

Obtaining Your Personal License Password (PLP) or License File

To install this software, you need a Personal License Password (PLP) for PC or Macintosh standalone licenses, or a License File for PC network and UNIX licenses. The MathWorks automatically sends the PLP or License File via e-mail or fax to the registered end user and the system administrator listed for the license. Without the PLP or License File, your software cannot be installed.

If you do *not* have your PLP or License File, then you can do ***one*** of the following:

- If you have your MATLAB Access number, you can look up your PLP or License File under the MATLAB Access area at `www.mathworks.com`

  You can request your MATLAB Access number under the MATLAB Access Program area at `www.mathworks.com` or send e-mail to `access@mathworks.com`
- Send e-mail to Customer Service at `serivce@mathworks.com`
- Call Customer Service at 508-647-7000, Ext. 3 or call your local representative

PC Installation Issues

Ensuring Sufficient Space for FTP Installation

The MATLAB installation is performed from the area indicated by the Microsoft Windows 95 or Windows NT environment variable `TEMP`.

If you run out of space while installing MATLAB via FTP, reset the `TEMP` environment variable to an area with sufficient space. Note that for FTP installations you need approximately 180 MB of space. The entire set of The MathWorks products is downloaded to the area specified by the `TEMP` environment variable; the installation process then installs only the products you specify.

Running MATLAB as a Networked Application

The networked version of MATLAB uses the TCP/IP communications protocol. Please note that although MATLAB is designed to run on any network that supports TCP/IP, only the Microsoft network has been fully qualified and is therefore the only officially supported network.

Run AXDIST.exe for Network Installations

If you are performing a network installation, run `axdist.exe` from the `AXDIST` directory on the CD-ROM for the current version of the OLE `aut32.dll`.

For single-user installations you do not need to run `axdist.exe`.

Windows NT 3.51 No Longer Supported

To run MATLAB 5.2 and its associated products on a Microsoft Windows NT platform, you must have Windows NT 4.0 with Service Pack 3 (i.e., Windows NT 3.51 is no longer supported).

UNIX Installation

Modifying the lmboot Parameter

The table on pages 1-14 to 1-15 of the December 1996 edition of the *MATLAB Installation Guide for UNIX*, Version 5.0 includes instructions to "Fix username argument to lmboot_TMW5 in this file." This means, for example, if your user name is `Smith`, the line in the Bourne shell fragment would read as follows:

```
/etc/lmboot_TMW5 –u Smith && echo ' MATLAB_lmgrd'
```

Note that the May 1997 reprint of the *MATLAB Installation Guide for UNIX* was updated to include the above information.

Default Web Browser Is Netscape Navigator

The MATLAB Help Desk requires an appropriate Web browser (but not Internet access). The default browser is Netscape Navigator, as found on the UNIX path. For directions on how to change this default, at the MATLAB command line enter

```
help docopt
```

Macintosh Installation

The GUI for the Macintosh installation has been modified for MATLAB 5.2. The interface now uses check boxes, similar to the PC installation interface. The installation process itself, however, is the same as it was for MATLAB 5.1.

MATLAB 5.2 Documentation

Help Desk

The MathWorks Help Desk provides access to online help topics, online reference materials, electronic documentation, and World Wide Web pages through a Web browser. You do not need to be connected to the Internet to access the online documentation.

Windows and Macintosh users can access this facility via the **Help** menu or the **?** icon on the Command Window toolbar. Users on all platforms can access the Help Desk with the `helpdesk` command.

Supported Browsers

To run the Help Desk, you should use Netscape Navigator Release 3.0 or 4.0.4 (earlier releases of Netscape Navigator Version 4 do not work well with the Help Desk), or with Microsoft Internet Explorer 3.0 or 4.0.

Specify Web Browser in docopt.m File for UNIX

The MATLAB Help Desk requires an appropriate Web browser (but not Internet access). On UNIX platforms, you must specify the browser you want to use in the `docopt.m` file; that file includes detailed directions.

Full-Text Search Facility

The 5.2 Help Desk includes a full-text search facility for the HTML online documentation. You can access the full-text search facility from the top page of the Help Desk or from the “Search” link on reference pages.

Reference Page Navigation

The 5.2 HTML reference pages provide additional navigational aids. The “Examples” and “See Also” links at the top of the first reference page for a function allow you to jump directly to the examples or to links to associated functions.

Also at the top of the reference pages is a “Go to function” edit box. Enter the name of the function and press the **Enter** key to see the reference page for that function.

The doc Command

The doc command now accesses the HTML reference documentation for all MathWorks products for which HTML reference documentation has been installed. Before Version 5.2, the doc command only accessed the documentation for MATLAB functions.

MATLAB 5.2 Documentation Set

The MATLAB documentation set has been expanded and in some cases rewritten. The set consists of online help, as well as hypertext-based manuals.

If you are upgrading to MATLAB 5.2 from an earlier version of MATLAB, then you should read *MATLAB 5.2 Product Family New Features*. The *New Features* document describes the enhancements added with MATLAB 5.0 through MATLAB 5.2. That document provides a roadmap for which sections you need to read, based on the version of MATLAB from which you are upgrading.

If you are a new MATLAB user, you should start by reading *Getting Started with MATLAB*, which explains how to MATLAB fundamentals.

Manuals Reprinted for 5.2

The following manuals have been printed and distributed to existing customers of MATLAB and its optional associated products, as part of their update package:

- *Late-Breaking News for the MATLAB 5.2 Product Family*
- *Application Program Interface Guide*
- *MATLAB Compiler User's Guide*
- *MATLAB C Math Library User's Guide*
- *MATLAB C++ Math Library User's Guide*
- *Communications Toolbox New Features Guide (Version 1.3)*
- *Financial Toolbox User's Guide*
- *Fuzzy Logic Toolbox User's Guide*
- *Neural Network User's Guide*
- *Spline Toolbox User's Guide*

Manuals Updated Online for 5.2

All the updated manuals listed above are also available online, via the Help Desk, in PDF format. The *Late-Breaking News for the MATLAB 5.2 Product Family* and the reference sections of the documentation for most of these toolboxes and blocksets are also available in HTML form.

In addition, the following manuals have been updated for 5.2 in PDF form. The reference section of almost all these manuals is available in HTML format, too.

- *Using MATLAB*
- *Using MATLAB Graphics*
- *MATLAB Function Reference* (includes language and graphics)
- *Application Program Interface Reference*
- *MATLAB Notebook User's Guide*
- *MATLAB C Math Library Reference*
- *MATLAB C++ Math Library Reference*
- *Using Simulink*
- *Real-Time Workshop User's Guide*
- *Stateflow User's Guide*
- *Control System Toolbox User's Guide*
- *Financial Toolbox User's Guide*
- *Frequency Domain System Identification Toolbox User's Guide*
- *Image Processing Toolbox User's Guide*
- *Mapping Toolbox*
- *Partial Differential Equation Toolbox User's Guide*
- *Robust Control Toolbox User's Guide*
- *Signal Processing Toolbox User's Guide*
- *System Identification Toolbox User's Guide*
- *Wavelet Toolbox User's Guide*
- *DSP Blockset User's Guide*
- *Power System Blockset User's Guide*

Documentation Updates

gsvd Function Added

MATLAB 5.2 provides a new function, gsvd, for generalized singular value decomposition. The gsvd function is documented in the online *MATLAB Function Reference*, but was not highlighted in *MATLAB 5.2 Product Family New Features*.

PC and UNIX Usage Information

PC

See “PC-Specific Problems” on page 1-18 for information about known MATLAB software problems on PC platforms.

Printing Under Microsoft Windows

Before you can print from a Microsoft or Novell NetWare network environment under Windows 95 or NT, you must map the LPT1 port to the printer you want to use.

To map LPT1 on Microsoft networks, issue this command at the system’s command prompt:

```
net use LPT1: \\server\printer
```

where *server* is the name of the server sharing the printer and *printer* is the name of the printer.

On Novell NetWare networks, use this command:

```
capture l=1 q=printer
```

where *printer* is the name of the print queue.

If you are using a Microsoft network, you can map LPT1, or you can edit the `printopt` function to change the definition of `pcmd` to:

```
COPY /B %s \\server\printer:
```

where *server* is the name of the server sharing the printer and *printer* is the name of the printer.

Problems Printing Using Z-Buffer If your system takes an excessively long time to print Z-buffer figures, you may need to switch to painters mode in MATLAB. See the chapter on printing in *Using MATLAB Graphics.*

Notebook Support

For Office 97 on Windows 95. The MATLAB 5.2 Notebook is fully supported for Windows NT with Microsoft Office 97. However, for Windows 95, due to an Office 97 problem, printing a Notebook document that includes an imported graphic may not print correctly. See “OFF97: Imported EMF Files Are Not Printed Correctly” in the online Microsoft Knowledge Base for details.

Microsoft Word 6.0 No Longer Supported. You must use either Microsoft Word 95 (Word 7.0) or Word 97.

UNIX

UNIX Installation Messages

- When you install MATLAB on an HP700 running the HP-UX 10.01 operating system, the installation succeeds, but several error messages appear in the shell window at the end of the process:

  ```
  A fatal error occurred while running 'xsetup' the X Window System
  version of 'install'. The following error was returned by this
  program:
  Tar: blocksize = 16
  X Error of failed request: BadMatch (invalid parameter
  attributes)
  Major opcode of failed request:  42 (X_SetInputFocus)
  Serial number of failed request:  12029
   Current serial number in output stream:  12030
  ```

  You may safely ignore these messages.

- When you install MATLAB on any HP700 or SGI workstation, the installation succeeds, but you may notice a message in the shell window at the end of the installation containing the following line:

  ```
  Tar: blocksize = 16
  ```

  You may safely ignore this message.

Editor Options Stored in Registry File

The MATLAB Editor stores your options in a registry file located under the directory `$HOME/.windu`. Communication with this file is handled through a daemon (called `windu_registryd41` or `windu_registryd40`), which is started automatically when you start MATLAB, and is shut down shortly after you quit MATLAB.

You can view the registry by typing the command `regedit` at the MATLAB command line. The regedit command starts the registry editor. Your settings are located under `HKEY_CURRENT_USER/Software/MathWorks`.

This registry is not portable across different UNIX platforms. The registry can only be read and written to on the platform on which it was created. If you start the MATLAB Editor on a different platform, it will start a registry daemon with the default options. These options can be saved and retrieved from the registry daemon, but once the daemon is shut down, the options will be lost.

When you start the Editor, if you get messages about missing options, it is possible that your registry has somehow been corrupted. To fix this problem, kill your registry daemon and delete the registry database directory. Then restart MATLAB and it will create a new registry with the default options.

Help Desk Search on DEC Alpha Running Netscape Navigator

Netscape Navigator 3.0 does not correctly implement Java on the DEC Alpha platform. Therefore, the search utility of the MATLAB Help Desk does not run on DEC Alpha under the Netscape browser. You can use the MATLAB `lookfor` function to help search for command line help by topic.

Adobe Acrobat Reader Not Supported for Xoftware

The Adobe Acrobat Reader, which is required to access the PDF files available through the Help Desk, is not supported for Xoftware.

Compatibility Issues

The following minor compatibility issues are not included in the *MATLAB 5.2 Product Family New Features* document.

Use of P-Code Between MATLAB Versions

You cannot use Version 5.2 P-code in a pre-5.2 P-code application. You can use pre-5.2 P-code in a Version 5.2 P-code application.

If you want to distribute an application to users who might be running a different version of MATLAB than the one in which you are writing the application, you should use M-files instead of P-code.

Colon Expressions with Floating-Point Numbers

Values produced in colon (:) expressions may vary between MATLAB 5.2 and pre-5.0 versions of MATLAB, if you are doing an exact comparison of floating-point numbers.

For floating-point numbers, you should use tolerance-based comparisons (eps), not exact comparisons. (Use exact comparisons only for integers.)

Warning When Using == with an Empty Matrix

The expression `A == []` produces `0` or `1` (as it did in MATLAB 4), and MATLAB issues the following warning message when this expression is used:

```
Warning: X == [] is technically incorrect. Use isempty(X) instead.
```

This warning is issued in anticipation of future versions of MATLAB, which will return an empty matrix, `[]`, for this expression.

Invoking the Path Editor from the Command Line

To invoke the MATLAB path editor from the command line on Microsoft Windows 95 or NT, UNIX, or Macintosh platforms, issue the `pathtool` command. In previous releases on various platforms the `pathedit` and `editpath` commands also invoked the path editor, but the command that works on all platforms for Version 5.2 is `pathtool`.

Frame Uicontrols and Stacking Order

Frames are opaque, not transparent, so the order you define Uicontrols is important in determining whether Uicontrols within a frame are covered by the frame or are visible. *Stacking order* determines the order objects are drawn: objects defined first are drawn first; objects defined later are drawn over existing objects. If you use frames to enclose objects, you must define the frames before you define the objects.

Before MATLAB 5.2, frames were always drawn below other Uicontrols on Microsoft Windows applications regardless of the order they were created.

If you use MATLAB on UNIX or Macintosh computers, this change does not affect you. If, however, you use MATLAB on Microsoft Windows, stacking order affects any applications that define frames *after* they define objects contained within the frames. To ensure that frames are drawn *below* other objects, either:

- Revise the M-files by altering the order in which these objects are defined. Create frames before creating the objects contained in the frames.
- Modify the stacking order to ensure that objects within the frames are visible. For example, these statements define a push button, a check box, and a frame, then alter the stacking order for the Figure (`position` vectors are defined by `pbpos`, `cbpos`, and `fpos` to simplify the code):

  ```
  hpush  = uicontrol('Style','pushbutton','Position',pbpos);
  hcheck = uicontrol('Style','checkbox','Position',cbpos);
  hframe = uicontrol('Style','frame','Position',fpos);
  % change stacking order to put frame on bottom
  % gcf is the current figure
  stackvec(1)=hpush; stackvec(2)=hcheck; stackvec(3)=hframe;
  set(gcf,'Children',stackvec)
  ```

- Issue a `system_dependent` command to force frames to be drawn below other objects. The form of this command is:

  ```
  system_dependent('ForceFramesOnBottom','on')
  ```

  Note that the `ForceFramesOnBottom` string is case sensitive. Issue the command before running the application. When you issue the command, MATLAB issues a warning indicating that frames will be inserted below other objects. To suppress the warning message for just this command, include these statements in your M-file or your `startup.m` file:

```
warning off
system_dependent('ForceFramesOnBottom','on')
warning on
```

You should use these commands only until you have had a chance to correct the M-files. The first two solutions are preferable to this solution; this solution is provided to ease the transition for users who were not aware that the Microsoft Windows behavior was inconsistent with stacking order rules that applied to all other Handle Graphics® objects.

You can turn off this behavior using this statement:

```
system_dependent('ForceFramesOnBottom','off')
```

PC-Specific Changes

Change to clc Command

In MATLAB 5.2, the `clc` command produces the same result as using the **Edit** menu item **Clear Sessions**. Thus, after you issue `clc`, you can no longer scroll back to see the previous contents of the Command Window (as you could in earlier versions of MATLAB).

However, you can use the up arrow to see the history of the commands, one at a time.

Change to cd Command

In MATLAB 5.2, if you `cd` from one drive to another for your working directory, the `cd` command does not retain any subdirectory part of the path if you `cd` back to the initial drive.

For example, if you first issue a `cd` command such as

```
cd C:\MyApps
```

and then issue

```
cd D:\MyMatlabDir
cd C:
pwd
```

you will see

```
C:\
```

In earlier versions of MATLAB, if you issued the same commands as shown above, you saw

```
C:\MyApps
```

Known Software Problems or Limitations

This section describes MATLAB 5.2 known software problems, providing workarounds for most problems.

Editor/Debugger

Opening Multiple Arrays in the Array Editor

If you select more than one array in the Workspace Browser and click **OK** to open them, the Workspace Browser (and possibly other GUI tools) will crash. To avoid this problem, open arrays one at a time.

Graphics

Note that there are some additional graphics issues discussed in the section "PC-Specific Problems" on page 1-18 and "UNIX-Specific Problems" on page 1-20.

Tooltips Do Not Work on Windows 95 with comctl32.dll Version 4.00.950

The tooltip feature introduced in MATLAB 5.2 does not work with a certain configuration of Microsoft Windows 95 because of a bug in that operating system.

In particular, tooltips do not work on computers having version 4.00.950 of `comctl32.dll`. To determine which version of `comctl32.dll` you have, run the Windows Explorer. In the `Win95` directory, display the files in the `System` directory. Right-click on `comctl32.dll` and select **Properties**. Then, select the **Version** tab to see the version number.

Internet Explorer version 3 upgrades `comctl32.dll` to version 4.70; Internet Explorer version 4 upgrades the file to version 4.71. Tooltips work with either version. Also, more recent versions of Windows 95 correct this problem. You can access the complete self-extracting archive file `Com32upd.exe` at

```
http://support.microsoft.com/download/support/mslfiles/Com32upd.exe
```

Note that even though your computer may have an upgraded version of `comctl32.dll`, if you are writing an application to be used by other users, they will not be able to take advantage of tooltips if their computers have the older version of the file.

Printing GUIs

GUIs that have a very large number of Uicontrols may take a long time to print when you use the `print –dwin` command.

Additionally, if you open another application on top of the GUI window while printing is in progress, elements of the top window may appear in the printout.

A warning is produced. To avoid this, wait until MATLAB indicates printing is finished before opening another window on top of your GUI.

Printing with the HPGL Driver

The HPGL `-dhpgl` print driver does not perform clipping in MATLAB 5.2. As a result, the text labeling port names in Simulink blocks can extent beyond the rectangle enclosing the block. Also, zooming in on line plots can produce lines that are not clipped to the Axes box, (i.e., it looks as if the Line objects Clipping properties are set to `off`).

Note that Microsoft Word users who import HPGL illustrations into documents will see no difference (since Word does not honor clipping).

dragrect and rbbox Functions

The `dragrect` function assumes that rectangles passed to it are specified in pixel units, not current Figure units. To avoid incorrect scaling of the rectangle(s) when using `dragrect`, make sure that arguments are specified in pixel units.

In function `rbbox(initialRect,fixedPoint,stepSize)`, the `stepSize` argument is in pixel units. Arguments `initialRect` and `fixedPoint` are in the current Figure units.

MinorGridLine Style Property Not Supported

The Axes `MinorGridLineStyle` property, which is mentioned in Chapter 10 of the MATLAB 5.0 *Using MATLAB Graphics* manual and returned when you query Axes properties, is not currently fully supported.

Application Program Interface

Avoid Modifying Input Arguments in MEX-Files

In MATLAB 5.1, MATLAB arrays can share data. There is currently no way for a MEX-file to determine that an array contains shared data. MEX-files that modify their input arguments may corrupt arrays in the MATLAB workspace. This style of programming is strongly discouraged.

Fortran Interface Not Implemented for New API Routines

The Fortran interface for API routines introduced in Version 5.0 is not implemented. The Fortran interface is only for routines in MATLAB 4.

Engine Support

Engine support is now implemented on PC platforms using ActiveX. Only V4 data types are currently supported.

Compiling with the Watcom 11.x Compiler

On the PC, when compiling MEX-files with Watcom 11.x, it is necessary to add the `watcom\binnt` directory to your DOS path.

Powerstation Fortran No Longer Supported

Powerstation Fortran 4.0 is no longer supported for MEX-files. DEC Visual Fortran 5.0 is supported.

PC-Specific Problems

The following problems apply to Microsoft Windows 95 and Windows NT platforms running MATLAB, unless otherwise indicated.

Netscape Navigator

If you try to access (e.g., via the helpdesk command or the Simulink **Help** button) the HTML online documentation via the Netscape Navigator browser and the documentation does not appear, open the browser manually and then access the online documentation. In general MATLAB will automatically start the browser for you, but in some Netscape Navigator installation configurations that will not occur.

Launching the M-File Editor/Debugger Without MATLAB

When you launch the M-File Editor/Debugger without MATLAB open, it becomes a pure editor; you cannot use it as a debugger.

To be able to use both the editing and debugging capabilities, have MATLAB open when you launch the M-File Editor/Debugger from Microsoft Windows, or start the M-File Editor/Debugger from within MATLAB.

Cannot Set Breakpoint in an M-File Shadowed by a P-File

When a P-code file, say `foo.p`, has been generated for an M-file, `foo.m`, via the `pcode` command, you can no longer debug the `foo` function graphically. If you set a breakpoint from the Editor/Debugger, the breakpoint icon will appear, but execution won't stop at the breakpoint when you run `foo`. If you set a breakpoint from the command line, the breakpoint icon won't appear but execution will stop at the breakpoint.

To work around this limitation, use *either*

- Command line debugging
- Graphical debugging, setting breakpoints from the command line and tracking them using `dbstatus`

Browsing Paths

When you click on the **Browse** button in the Path Browser, the **Change Current Directory** dialog may not default to the current directory. If you have a path highlighted in the **Path** listing when you click **Browse**, that path is displayed in the **Change Current Directory** dialog. To start browsing from your current directory, click in the **Current Directory** box before clicking the **Browse** button.

Avoid Spaces in Directory Names and Filenames

To ensure predictable results when running MATLAB on the PC, avoid using spaces in directory names and filenames for the locations you use for installing MATLAB and its ancillary products such as the compilers used for building MEX-files.

Following this recommendation shields you from a limitation in the Windows 95 and Windows NT command environment. If you do use spaces in such directory names or filenames, you may encounter problems stemming from the

Windows command environment, such as messages about being unable to find files that are actually where you think they are.

UNIX-Specific Problem

Cutting and Pasting from X Window Systems Applications

You cannot use the Editor on UNIX platforms to cut and paste to and from X Window Systems applications.

Editor Does Not Support Japanese Characters

The Editor on UNIX platforms does not accept Japanese characters as input.

MATLAB Compiler 1.2

Known Software Problems and Limitations

This section describes known software problems and limitations for the MATLAB Compiler 1.2.

Compatibility Release

Version 1.2 of the MATLAB Compiler is a compatibility release that brings the MATLAB Compiler into compliance with MATLAB 5. Although the Compiler works with MATLAB 5, it does not support many of the new features of MATLAB 5. Specifically unsupported are:

- Cell arrays
- N-D arrays
- Structures
- Objects
- Variable number of arguments
- `switch` and `case` statements

Some of the restrictions in Version 1.1 of the Compiler have been removed or changed for Version 1.2. For a complete list of restrictions, see the *MATLAB Compiler User's Guide*, Version 1.2.

Watcom 10.6 Compiler

If you are using Watcom 10.6, you cannot install MATLAB in a directory whose name contains special characters such as dashes or dollar signs. For example, installing MATLAB in a directory named `matlab5-1-c` would cause a failure because Watcom 10.6 truncates this directory name, causing problems when you try to build MEX-files.

Linking MFC Applications with the C++ Math Library

In order to allow Microsoft Foundation Class (MFC) applications to link against the C++ Math Library, the MSVC version of the library is built against the multithreaded DLL versions of the MSVC runtime libraries. The impact of this change is as follows:

Application	Change Required
`mbuild`	Works the same.
MFC applications in the Visual C++ IDE	Use MFC as a DLL (the default).
MSVC Console applications in the Visual C++ IDE	Change the project settings for C/C++; go to the **Project settings** dialog, choose the **C/C++** tab, then the **Code generation** item, and select either **Multithreaded DLL** or **Debug Multithreaded DLL** for **Runtime Library**.
Building from command line without `mbuild`	Use `–MD` switch when compiling.

Additionally, the MSVC runtime libraries, `msvcrt.dll` and `msvcirt.dll`, need to be distributed with any applications created with the C++ Math Library on Visual C++.

Runtime Errors Involving Operands

The Compiler may not always detect runtime errors involving incorrectly typed operands. If you suspect that an error might be occurring within your code, check the MATLAB version first. If the MATLAB version produces no errors, then the Compiler-generated code will work correctly.

ode Functions on 68K Macintosh

The 68K version of `libmmfile` does not contain the ode functions `mlfOde113`, `mlfOde15s`, `mlfOde23`, `mlfOde23s`, and `mlfOde45`. These functions are unavailable on the 68K platform.

Documentation

The *MATLAB Compiler User's Guide* has been updated to reflect the current version of the Compiler. See `<matlab>/toolbox/compiler/Readme.m` for additional details about using the Compiler.

3

Simulink 2.2

Known Software Problems and Limitations

Help Button

Under Microsoft Internet Explorer 4, the Simulink HTML online documentation does not display if invoked via a **Help** button on a Simulink dialog box in the case where the online doc is installed on a network drive separate from where your `temp` directory is located. This is due to new cross-frame security restrictions in Internet Explorer 4. You can, however, access the same information through the Help Desk.

S-Functions

You must recompile any S-functions written using a version of Simulink® 2.0.

Any S-function that indicates it has no inputs and/or outputs causes Simulink to remove the corresponding port or ports from the S-Function block icon. This behavior is different than in Simulink 2.0.

The ability to use a block diagram as an S-function is not implemented in Simulink 2.0. There is no workaround; work is in progress to reimplement this functionality or an equivalent for a later release.

Status Bar

The status bar does not accurately show the elapsed simulation time and progress bar.

Using an MGA Matrox Board

If your computer has an MGA Matrox board, you might find that some circles (Simulink block outlines) are drawn in the wrong window when opaque dragging is in effect and you drag a window over a Simulink model window. If this is the case, right-click on the desktop, select the **Properties** item from the pop-up menu, and then the **MGA Settings** tab. Select the **Advanced** option, and then the **Performance** tab. Do *not* select the **Circle and Ellipse Acceleration** option.

Microsoft Windows and Lost Windows

Microsoft Windows users may find that an open window does not appear on the desktop because that window's position is beyond the bounds defined by the size of the monitor. You can display all open windows by selecting an option from the bar that appears along the bottom of the desktop. Right-click either on an unoccupied area or on the clock on that bar. Then, select either **Cascade**, **Tile Horizontally**, or **Tile Vertically**. These options display all windows on the desktop.

Printing on Microsoft Windows 95

On Microsoft Windows 95, Simulink's **Printer Setup** dialog fails to set the paper orientation for printing block diagrams. To set paper orientation, select **Print...** from the **Simulink File** menu and then select "Properties" on the resulting **Print** dialog.

Real-Time Workshop 2.2

The Real-Time Workshop 2.2

Summary of Enhancements Introduced in 2.1

If you are upgrading from Real-Time Workshop® 1.1 to 2.2, note the enhancements that were added in Version 2.1 are also included in Version 2.2. These features are documented in the online version of the *Real-Time Workshop User's Guide*.

Architectural Changes

The structure of Version 2.1 of the Real-Time Workshop differs significantly from Version 1.1. See Chapter 6 of the *Real-Time Workshop User's Guide* for more information about the architecture of the Real-Time Workshop; read that chapter if you are interested in how the Real-Time Workshop generates C code.

Target Language Compiler

Part of the Real-Time Workshop's new functionality is a reliance on the Target Language Compiler™. For information on what this is, how it works, and how to customize it to your needs, refer to the *Target Language Compiler Reference Guide*.

Generic Real-Time Target

This release works with fixed-step models that are designed to model real-time systems. A generic real-time target is provided for performing high-speed simulations on your workstation. This target is also a useful example of how to target custom hardware.

Additional Changes

Changes from Version 1.1 include:

- The user interface now uses appropriate Simulink model settings.
- The external mode, `ext_comm.mex` file provided by The MathWorks for external mode is now based upon a TCP protocol.

Unsupported Features in Version 2.2

Version 2.2 of the Real-Time Workshop does *not* currently support:

- Nonreal-time simulation of variable-step solvers
- The Simulink Accelerator

Upgrading External Mode MEX-Files

You must modify existing external mode link MEX-files slightly and rebuild them.

You must replace the structure definition of `ExternalSim_tag` and `#defines` for `EXT_x` names with:

```
#include "extsim.h"
```

Note: When you call `mdlCommInitiate`, Real-Time Workshop automatically follows that call with a call to `mdlSetParameters` to download the initial set of parameters.

For more information, see *matlabroot*`/rtw/ext_mode/ext_tmpl.c` or *matlabroot*`/rtw/ext_mode/ext_comm.c` for a TCP socket-based implementation.

The Real-Time Workshop and Stateflow

If you add Stateflow™ blocks to your Simulink model, the Real-Time Workshop directs Stateflow to generate two additional files. These files are:

- *model*`_rtw.c`
- *model*`_sfun.c`

Tornado/VxWorks

When you use the StethoScope Real-Time Graphical Monitoring/Data Collection Utility (`STETHOSCOPE=1` in `make` command arguments), the build procedure now automatically loads the required StethoScope libraries onto the target system. For more information, see the Tornado template makefile, *matlab_root*`/rtw/c/tornado/tornado.tmf`.

There are three functions supplied in *matlab_root*/rtw/c/tornado/rt_main.c that you can call from a **Wind Shell** (windsh) window to allow the user to dynamically select signals to be monitored by StethoScope while the real-time simulation is running. To install or remove block outputs in StethoScope interactively, use rt_installSignal and rt_removeSignal, respectively. To print a list of block names that consist of a given string, use rt_lkupBlocks. See rt_main.c for arguments required for these functions.

Note About Automatic Downloading

If the target system is rebooted, the Tornado Target Server and the Tornado Registry may need to be restarted. This can be tested by seeing if the Wind Shell (windsh) can still connect to the target when run from the command line.

Target Language Compiler (TLC) Enhancements

This section describes enhancements to the Target Language Compiler that were not documented in the *MATLAB 5.2 Product Family New Features* guide.

Passing Parameters: mdlRTW and RTWData

The Real-Time Workshop generates a `model.rtw` file that is a description of the model. There are two additional methods of passing user-specified information into the `model.rtw` file:

- `mdlRTW` — Used with Level 2 S-functions
- `RTWData` — Used with any nonvirtual Simulink block and with empty subsystems

mdlRTW

Level 2 S-functions can use the `mdlRTW` function to pass information from a C-MEX S-function into the `model.rtw` file for use during code generation.

The information that the `mdlRTW` function writes to `model.rtw` is used by the block target file for that block type. The writer of the block target file can use the additional identifier/value pairs as desired. For all the possible functions that you can use inside `mdlRTW` to generate information in the `model.rtw` file, see the file *matlabroot*`/simulink/src/sfuntmpl.doc`. See Chapter 8 of *Using Simulink* (online version) for a discussion of how to write an `mdlRTW` function.

Example code. This is an example of how to use `mdlRTW` in a Level 2 S-function:

```
static void mdlRTW(SimStruct *S)
{
    int_T numElements = mxGetNumberOfElements(TASK_NAME);
    char *buf = NULL;

    if ((buf = malloc(numElements +1)) == NULL) {
        ssSetErrorStatus(S,"memory allocation error in mdlRTW");
        return;
    }
    if (mxGetString(TASK_NAME,buf,numElements+1) != 0) {
        ssSetErrorStatus(S,"mxGetString error in mdlRTW");
        free(buf);
        return;
    }
    /* Write out the parameters for this block.*/
    if (!ssWriteRTWParamSettings(S, 3,
         SSWRITE_VALUE_QSTR,"TaskName", buf,
         SSWRITE_VALUE_NUM,"Priority",
         (real_T) (*(mxGetPr(PRIORITY))),
         SSWRITE_VALUE_NUM,"StackSize",
         (real_T) (*(mxGetPr(STACK_SIZE))))) {
      return; /* An error occurred which will be reported by SL */
    }

    /* Write out names for the IWork vectors.*/
    if (!ssWriteRTWWorkVect(S, "IWork", 1, "TaskID", 1)) {
      return; /* An error occurred which will be reported by SL */
    }

    /* Write out names for the PWork vectors.*/
    if (!ssWriteRTWWorkVect(S, "PWork", 1, "SemID", 1)) {
      return; /* An error occurred which will be reported by SL */
    }

    free(buf);
}
```

This code contains the resulting model.rtw information:

```
Block {
  . . .
  SFcnParamSettings {
                TaskName""
                Priority20
                StackSize1024
  }
  NumIWorkDefines      1
  IWorkDefine {
                Name         "TaskID"
                Width        1
  }
  NumPWorkDefines      1
  PWorkDefine {
                Name         "SemID"
                Width        1
  }
}
```

RTWData

RTWData is a parameter that you can set on Simulink blocks using the set_param() command and view with the get_param() command. The parameter/value pair is saved along with the model.

The command syntax is

```
set_param(gcb,'rtwdata',userdata)
```

where gcb is the current block pathname. The variable userdata must be a MATLAB data structure where each element is a string. For example:

```
userdata.a = 'rpm'
userdata.b = '1.25'
```

When attached to a nonvirtual block, the associated `model.rtw` information for the block is:

```
Block {
. . .
  RTWdata {
    a    "rpm"
    b    "1.25"
  }
}
```

The block target file for that block type can process the information as desired. For example, if `RTWData` is attached to a S-function, the TLC inlining file for the S-function could process the information in the `BlockInstanceSetup` function.

Besides nonvirtual blocks, `RTWData` can be attached to one special case of a virtual block, an empty subsystem. This allows information the be passed into the `model.rtw` without it being associated with a specific nonvirtual block. This is useful when some block-independent information needs to be passed into `model.rtw` for use during code generation. For empty subsystems, the `RTWData` parameter is placed in the `System` record for the nonvirtual system in which the empty subsystem is contained.

```
System {
. . .
  EmptySubsysInfo {
    NumRTWdatas    1
    RTWdata {
      a    "rpm"
      b    "1.25"
    }
  }
}
```

Because the empty subsystem technique is used by the Custom Code block of the `RTWLib`, there is support built into the system target files to handle `RTWData` attached to empty subsystems. Specifically, if an `EmptySubsysInfo` record exists, all `RTWdata` subrecords are checked for the existence of an identifier named `TLCFile`. If the identifier exists, the value of `TLCFile` is used as a block target filename and the TLC function `ProcessRTWdata` in that file is called using the `TLC GENERATE` directive. This functionality can also be used by other (user-written) blocks if desired.

TLC Documentation Updates

These changes were not incorporated into the *Target Language Compiler Reference Guide* distributed with Real-Time Workshop 2.1.

Some `model.rtw` files, such as the one appearing in Appendix A of the *Target Language Compiler Reference Guide*, caused difficulty when the signals were very wide. Compilation also took too long and required excessive memory to process adequately. To address these problems, several extensions to the Target Language Compiler were made in the area of vector handling. The Real-Time Workshop files have been modified to take full advantage of these new constructs.

Changes to Target Language Values Table

Several changes to Table 2-2, Target Language Values, in the *Target Language Compiler Reference Guide* are necessary.

1. The `"Range"` value type should specifically say that range values can only appear within vectors. The `"Vector"` value type is defined as being lists of values. The individual elements of a vector do not need to be the same type, and can be any type except vectors or matrices.

2. A new value type, `"Repeat,"` repeats the first value the number of times of the second value. For example, `[3@10]` is equivalent to `[3 3 3 3 3 3 3 3 3 3]`.

   Note: This can only appear within the context of a vector. See the “Target Language Vectors” section for information on how to interpret vectors containing repeated values.

3. Another new value type, `"Idrange,"` represents a range of identifier values where the base name is the leading identifier and the numeric range describes the values that it may take on. For example, `[B0:B20]` represents `[B0, B1, B2, ..., B20]`.

   Note: This can only appear within the context of a vector. See the “Target Language Vectors” section below for information on how these values are interpreted within vectors.

Target Language Vectors

Note: This is a new section, not an update to an existing section in the *Target Language Compiler Reference Guide*.

Target language vectors can take advantage of several short-hand notations that make vector specification easier. They can also improve performance and memory requirements of the TLC while it generates the code. The special value types "Range," "Idrange," and "Repeat" are interpreted by the Target Language Compiler as follows:

Value Type String	Example	Description
"Range"	[1:5]	Equivalent to placing all the numeric values within the range in the vector. For example, [1:5] is the same as [1, 2, 3, 4, 5]. Specifying ranges in this manner reduces the amount of memory required by the TLC and improves the TLC performance when translating the file. The first number in the range must be less than or equal to the second number.

Value Type String	Example	Description
"Idrange"	[B0:B5]	Equivalent to placing all of the identifiers within the range in the vector. For example, [B0:B5] is the same as [B0, B1, B2, B3, B4, B5]. Specifying identifier ranges in this manner reduces the amount of memory required by the TLC and improves the performance when translating the file. Both identifiers in the Idrange must be identical (e.g., B in this example) and the number in the first identifier must be less than or equal to the number in the second identifier.
"Repeat"	[3@5]	Repeats the specified value the indicated number of times. For example, [3@5] is the same as [3, 3, 3, 3, 3]. Specifying repeated values in this way reduces the amount of memory required by the TLC and improves the performance when translating the file. The second argument must be a number.

Changes to the SIZE Built-In Function

When calling the SIZE built-in function or specifying the vector size for the Target Language Compiler, you must count all elements expanded out in such a fashion. For example:

```
%assign x = Vector(5) [1@5]     %% This is correct; there are 5
                                %% elements in the Vector.
```

The SIZE function returns 5 for the vector x, and 5 must be specified if you use the Vector(#) syntax. When indexing into a vector, likewise, all of the elements are returned as if they had been enumerated.

Changes to Configurable RTW Variables Table

Several new variables have been added to the Configurable RTW Variables table in Chapter 3, “Writing Target Language Files.” The variables are:

```
ForceParamTrailComments
BlockIOSignals
ParameterTuning
```

ForceParamTrailComments

The trailing comments for each parameter value in `model.prm` are not generated when the number of parameters in the model is greater than 1000. Setting `ForceParamTrailComments` to 1 overrides this behavior.

BlockIOSignals

Setting `BlockIOSignals` to 1 generates the block I/O signals into `model.bio`. See the *Real-Time Workshop User's Guide* for details.

ParameterTuning

Use `ParameterTuning` to generate parameter mapping information when you want to modify parameters separately from Simulink's external mode. Setting this variable to 1 generates the parameter tuning information file, `model.pt`. For details see the file `MATLAB_ROOT/rtw/c/src/pt_readme.txt`. Note that this is an undocumented feature provided for your convenience, and that The MathWorks reserves the right to change the parameter tuning information file in subsequent releases.

Changes to Appendix A, model.rtw

Several of the parameters described in the `model.rtw` file have been modified.

SignalSrc

If needed, `SignalSrc` uses the `Range` (`:`) and/or `Repeat` (`@`) values. The `SignalSrc` parameter can be found in:

- `Signal` record of either the `RootSignals` or `Subsystem` record.
- `DataInputPort` record of a `Block` record.

SigConnected

SigConnected is now written as all, none, or a vector of ones and zeros. This parameter can be found in the BlockOutput records.

BlockOutputsMap

BlockOutputsMap is now a vector that uses the Range (:) and/or Repeat (@) values. For a given index into the block I/O vector (Bi), this gives the index of the BlockOutput record to which Bi maps. The offset within the record is computed by: Bi – SigIdx[0]

ExternalInputsMap

ExternalInputsMap is now a vector that uses the Range (:) and/or Repeat (@) values. For a given index into the external input vector, Ui, this gives the index of the ExternalInput record to which Ui maps. The offset within the record is computed by: Ui – SigIdx[0]

SignalSrcTID

SignalSrcTID uses the Repeat (@) value when possible.

BlockMap

Within the RootSignals or Subsystem records, BlockMap uses the Range (:) value when possible.

Signal Records

In the Signal records, if Block is a string, then it will optionally be followed by the parameter:

> SLBlock — Unmodified Simulink name. This is only written if Block is a string and is not equal to the block name.

Any SigLabel parameter will optionally be followed by:

> SLSigLabel — Unmodified Simulink label. This is only written if SLSigLabel is not equal to SigLabel.

The Signal records also contain the following parameters:

- OutportName — Only written for subsystem blocks. This is the name of the corresponding outport block.
- SLOutportName — Unmodified Simulink outport block name. This is only written if SLOutportName is not equal to OutportName.

ChildSubsystemIndices

ChildSubsystemIndices is now present in the RootSignals and Subsystem records. This is a vector that gives the subsystem record index for any child subsystems.

StatesMap

StatesMap is a matrix of dimension (NumContStates + NumDiscStates, 3).

New Error Messages for Appendix B

The second value in a range must be greater than the first value.

In valid ranges, the first value in the range must be greater than the second value. For example, [1:0] generates this error.

Invalid identifier range, the leading strings "<identifier>" and "<identifier>" must match.

When specifying an identifier range, the leading identifiers must match for both ends of the range. For example, [B0:U2] generates this error.

Invalid identifier range, the lower bound (<number>) must be less than the upper bound (<number>).

When specifying an identifier range, the lower bound number must be less than or equal to the upper bound number.

5

Stateflow 1.0.6

Introduction

This chapter provides information about Stateflow 1.0.6, which is being shipped with MATLAB 5.2.

Note: The version of Stateflow (1.0.6) shipped with MATLAB 5.2 is essentially the same as the Patch Release 1.0.5 recently made available via FTP to Stateflow customers. However, this version fixes some bugs that still existed in the patch release.

Product Requirements

Stateflow is a multiplatform product, running on Microsoft Windows 95, Windows NT, and UNIX systems.

Stateflow version 1.0.6 requires:

- MATLAB 5.2
- Simulink 2.2
- A C or C++ compiler for creating MATLAB MEX-files on your platform. For a list of all the compilers supported by MATLAB, see The MathWorks Technical Support Department's Technical Notes at:

```
http://www.mathworks.com/support/tech-notes/#mex
```

See the *MATLAB Application Program Interface Guide* for information on how to ensure proper installation for integration into the MATLAB environment.

In addition, you can optionally use the Real-Time Workshop to generate floating-point code for Simulink portions of the model.

Enhancements

Bug Fixes to Stateflow 1.0

Version 1.0.6 fixes these bugs that were in Version 1.0:

- Stateflow 1.0 failed to generate variable declarations when generating code for a Stateflow diagram that had no states and a data object parented by the chart with a **Temporary** scope. As a result, the generated code may not have compiled.
- Stateflow 1.0 generated a ratio of integers as the division of an integer constant by another integer constant, leading to incorrect results when compiled. For example, the expression `a = 1.0/2.0` translated to `a = ½` in the generated code. The compiler, using integer arithmetic, set a equal to `0` instead of the intended `0.5`.
- Using Kanji fonts caused segmentation faults when editing multiline labels.
- Workspace data with `min` and `max` values that were not `–Inf` and `Inf`, respectively, resulted in generated code that did not compile.
- Code generation required an `sfc.m` file.
- Using the Finder caused assertion failures under certain circumstances.
- Using the Simulink drop-down menu to copy and then paste a Stateflow block sometimes resulted in a warning and an error after pasting.

Enhancements to the Sensor Failure Detection Demo

The Sensor Failure Detection demo in Stateflow 1.0.6 includes these enhancements:

- The `fuelsys` model uses a new organization of the model objects, clarifying the interaction between the control logic and the engine model.
- Both the physical model of the engine system and the control logic have been refined to better represent engine behavior.
- The control logic algorithm provides improved control over the `air/fuel mixture ratio`. The `air/fuel mixture ratio` is monitored and displayed during simulation.

- A response to speed sensor failure has been added to the `Overspeed` state in the control logic.
- The Stateflow diagram update method has been changed from **Triggered** to **Sampled**, simplifying the Simulink and Stateflow interaction.

Current Restrictions

Supported C Compiler

A C or C++ compiler is required for creating MATLAB MEX-files on your platform. For an up-to-date list of compilers supported by MATLAB, see The MathWorks Technical Support Department's Technical Notes at:

```
http://www.mathworks.com/support/tech-notes/#mex
```

Simulink Libraries and Stateflow Blocks

Stateflow blocks do not currently support Simulink libraries.

Machine Properties Dialog Box

There is a **Machine properties** dialog box for each machine. Currently this dialog box is accessible only through the **Chart properties** dialog box. The machine is shown in the **Chart properties** dialog box as the chart's parent. Clicking on the **Parent** field hypertext link displays the **Machine properties** dialog box. You can specify these fields in the **Machine properties** dialog box:

- **Creator**
- **Modified**
- **Version**
- **Description**
- **Document Link**

Stateflow Finder and Case-Sensitive Searches

The Stateflow Finder performs case-sensitive searches based on the string criteria you enter. Currently you cannot specify a case-insensitive search.

Spaces in State Names

Spaces in the state name portion of a state label are not currently supported. For example, if you entered `state one` as one state's name and `state two` as another state's name (at the same level in the hierarchy), the parser would generate an error indicating the names are not unique.

Undo Operation

Stateflow does not currently support an undo operation. However, delete operations copy the deleted objects to the paste buffer. You can undo a delete operation by immediately pasting the deleted objects.

Directed Event Broadcast Using send

As described in the *Stateflow User's Guide*, you can specify a directed event broadcast in the action language. The format of the directed broadcast is:

```
send(event_name,state_name)
```

where `event_name` is broadcast to `state_name` and any offspring of that state in the hierarchy.

Currently, you cannot direct the `event_name` to a `state_name` in another machine.

Sharing Workspace Data in a Real-Time Workshop Target

Sharing workspace data between two or more Stateflow charts in a Real-Time Workshop (RTW)target can cause a collision when these conditions are present:

- The RTW target is multitasked.
- At least one of the charts has write access to the workspace data object.
- The Stateflow charts that share the data have different sample rates.

It is important to note that all of these conditions must be present to cause a problem. If these conditions exist, you should use an explicit method within Simulink to exchange the workspace data between charts. For more information, see the "Models with Multiple Sample Rates" chapter of the *Real-Time Workshop User's Guide*.

Real-Time Workshop RealTimeMalloc Code Format Option

The Stateflow Coder does not support the Real-Time Workshop `RealTimeMalloc` code format option.

Graphics Editor Edit Menu and Text Labels

The **Cut**, **Copy**, and **Paste** menu items on the graphics editor **Edit** menu operate on Stateflow objects (states, transitions, and junctions). Text, such as a state or transition label, is not considered an object. Currently, when you edit a text label, the **Edit** menu is disabled during the edit operation.

Known Software Problems

Resizing the Graphics Editor Window

If you resize the graphics editor window by quickly dragging the lower right-hand corner, the window may snap back to its original size. This behavior occurs only on PC platforms when opaque sizing is enabled.

Stateflow Finder Dialog Box Cannot Be Resized

You cannot resize the Finder dialog box in this release.

Stateflow Quick Reference Guide

Graphics Editor

Window Navigation and Zooming:

Esc key– Activate the Simulink window containing the Stateflow diagram

Space bar – Resize the Stateflow diagram to fit within the current window

F key– Zoom in on the selected object

Object Selection:

Left mouse click - Select a single object

Shift key + left mouse clicks - Individually select multiple objects; toggle object selection

Left mouse click + drag – Drag selection rubberband; all objects entirely or partially within the rubberband are selected

Grouping:

Left mouse double click - Group all objects within selected state; toggle grouping on and off

Action Language

Transition Label:

```
Event_name [condition] {condition_action} / transition_action
```

State Label:

```
name/
entry:
during:
exit:
on event_name:
```

Reserved Words:
(abbreviation)

change (chg)
during (du)
entry (en)
exit (ex)
in
on
send
matlab (ml)

Operators:	+	addition	%%	modulus	--	decrement
	–	subtraction	|	or	~	not
	=	assignment	&	and	–	negation
	*	multiplication	++	increment		

Punctuation:	{}	condition action	,	statement separator (display value)
	( )	function arguments	;	statement separator
	[ ]	condition	:	keyword/action separator

Event Broadcasts:

event – Broadcast *event* to its parent and complete the resulting executions.

state.*event* – Broadcast *event* to its parent *state*; evaluate the state if it is active.

send(*state*,*event*) – Direct *event* to *state* and evaluate *state*, if active.

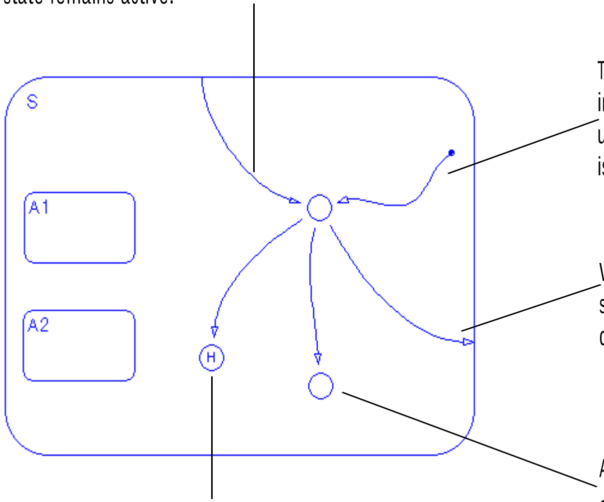
Function Calls:	ml.*name*(*arg1*, *arg2*, …)	MATLAB function call
	name(*arg1*, *arg2*, …)	Custom code function call
	name()	Function call with empty argument list

Flow Diagrams within States

Flow diagram notation is essentially logic represented without the use of states. Flow diagram notation eliminates the use of unnecessary states and produces more efficient code with optimized memory usage. Flow diagram notation is represented through combinations of transitions to and from connective junctions.

This diagram shows flow diagram notation used within a state:.

A transition originating on a state boundary will execute if the state remains active.



The default transition is executed when the state is initially entered. The default transition will execute upon subsequent entries to the state if no substate is specified.

When a transition terminates on the inner edge of a state, the next active state will be determined using default transitions or history.

A terminating connective is a connective with no outgoing transition. Terminating connectives stop path execution in the current state without processing any state `exit` or `entry` actions.

A terminating history junction stops the path execution on the active state. The active state will be exited and then entered. The execution of the state `exit` and `entry` actions distinguishes the behavior of history junctions from other terminating connectives.

Chart Execution

Event Processing

Stateflow processes one event at a time. When an event broadcast occurs, chart execution switches to the new event. The processing of an event involves two steps:

1 Searching for a valid transition path

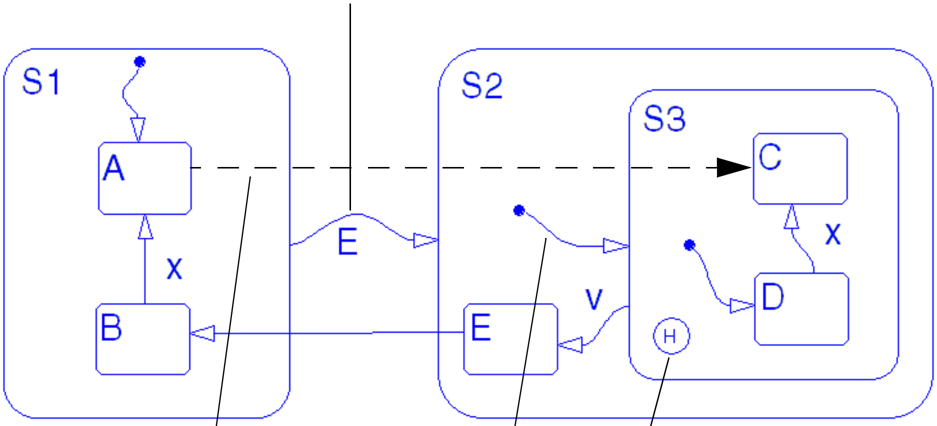
2 Executing the transition path

Valid Transition Paths

When an event occurs, Stateflow uses semantic rules to interpret the chart logic and to construct the resulting transition path. A valid transition path is an implied path between an active origin state and an active destination state. The implied path can contain one or more of the following: transitions, connectives, default transitions, history junctions, and implied transitions from a child to its parent.

This figure shows the transition path from state *S1.A* to state *S2.S3.C* when event *E* occurs:

Valid Transition



Resulting transition path
(not part of the original diagram)

Other objects that determine the path

Searching for a Transition Path

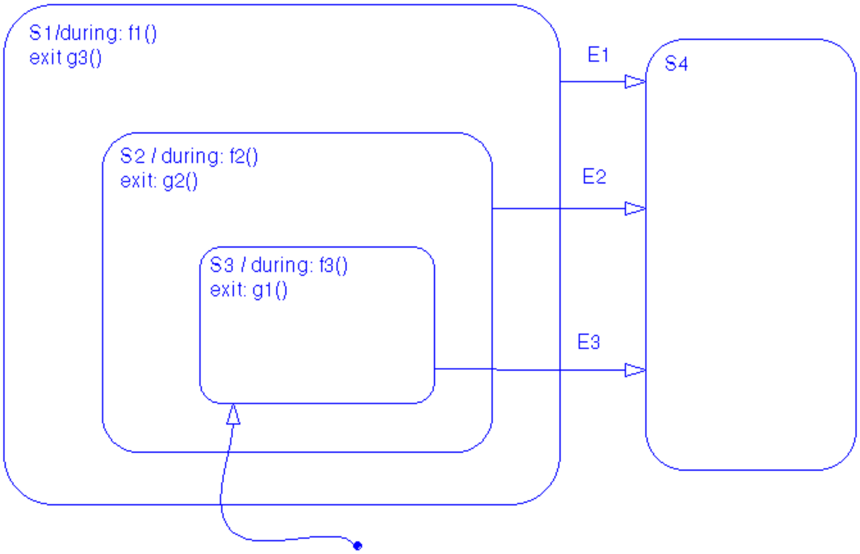
Stateflow uses a combination of hierarchy and ordering rules to determine the valid transition path. It is important to note that valid actions are executed during the path search.

The path search begins with the active state; all transitions that originate from the active state are tested. Ordering rules determine the sequence in which the transitions are tested. If a transition branches or loops through a connective, the outgoing transitions from the connective are tested. Searching is terminated when the first valid path is found.

Top-Down Searching. Searching starts at the outermost active state. The search steps include:

1. Test all outgoing transitions for a valid path.
2. Execute `during` action.
3. Execute `on event_name` action.
4. Test inner transitions for a valid path.

If these steps fail to produce a valid path, the next active substate in the hierarchy is activated and the search steps are repeated. As an example, consider the execution of this Stateflow chart:

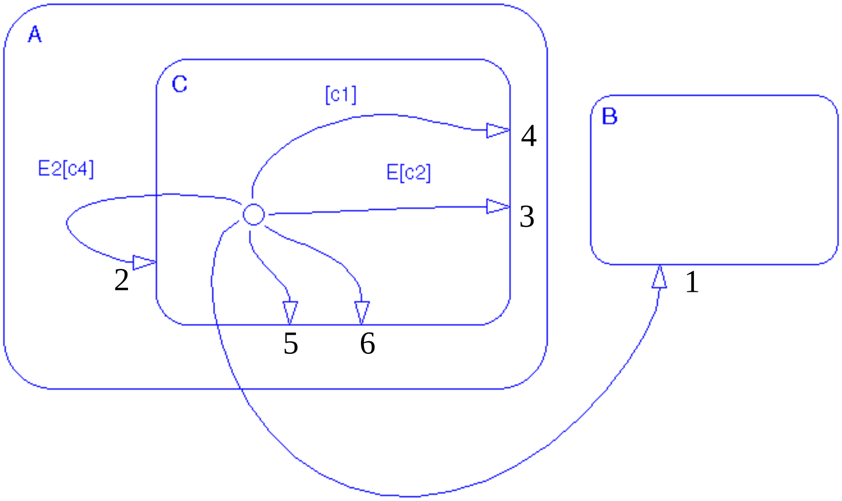


After the default transition executes, state S3 becomes active. The path out of state S3 is dependent on which event is broadcast:

Event Broadcast	Result
E1	No during actions are executed.
E2	The during action *f1()* is executed.
E3	The during actions *f2()* and *f3()* are executed.

A substate can be active only when its parent state is active. As a result, substates are entered last and exited first. Regardless of which event is broadcast in the chart on page 5-14, the exit actions will be executed in the order: *g1()*, *g2()*, and *g3()*.

Testing Order. Consider the group of transitions that originate from the connective in the diagram:



The numbers near the transition arrowheads indicate the sequence in which they are tested. This table provides the ordering rules that Stateflow uses:

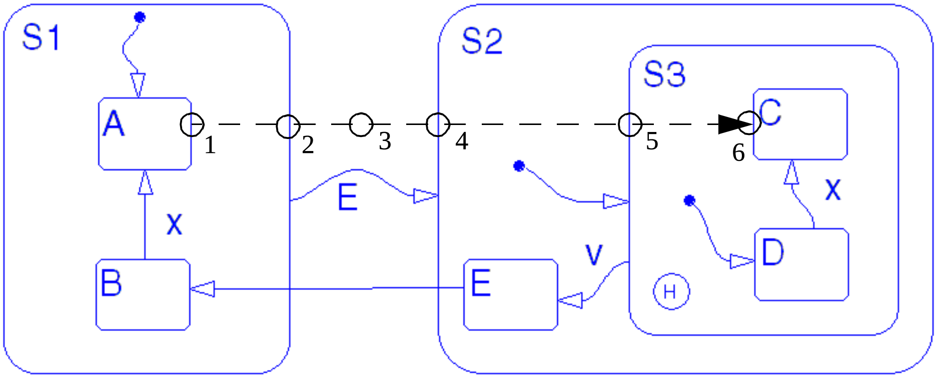
Rule Priority	Description	Transition from chart on page 5-15
1	**Parent Rule**: Transitions are sorted by the level of their parent. This means that outermost transitions are tested first.	1, 2
2	**Transition Label Rule**: Transitions with the same level of parent are sorted by their label. The labels are considered in the following order: **1** Labels that include events and conditions **2** Labels with events only **3** Labels with conditions only **4** Labels with no events or conditions	3, 4, 5, 6
3	**Transition Origin Rule**: Transitions with the same level parent and the same type of label are sorted by their origin point using object geometry. The rule applies to both state and connective origins: • For state origins, the sorting is clockwise starting at the upper left hand corner. • For connective origins, the sorting is clockwise starting at 12:00.	5, 6

Path Execution

As a valid transition path is executed, states are marked active or inactive. The sequence of marking a state active or inactive is equivalent to the order in which state boundaries are crossed with the path. In addition, state `entry` actions, state `exit` actions, and transition actions are executed as follows:

- State `exit` actions are executed just before a state is marked inactive.
- State `entry` actions are executed just before a state is marked active.
- Transition actions execute after all states have exited and before any state is entered.

This figure shows the diagram on page 5-13 with the addition of labels to mark critical points along the transition path:



Referring to these labels, consider the execution of the transition path from state `S1.A` to `S2.S3.C`:

1. The `exit` action of state `S1.A` executes and `S1.A` is marked inactive.

2. The `exit` action of state `S1` executes and `S1` is marked inactive.

3. All transition actions execute in the order in which they are encountered along the path.

4. State `S2` is marked active and its `entry` action is executed.

5 State `S2.S3` is marked active and its `entry` action is executed.

6 State `S2.S3.C` is marked active and its `entry` action is executed.

Frequently Asked Questions

This table contains frequently asked questions and their answers:

Questions	Answers
What are the differences among the terms Stateflow diagram, chart, and Stateflow block?	Using Stateflow, you create Stateflow diagrams. A Stateflow diagram is also a graphical representation of a finite state machine where states and transitions form the basic building blocks of the system. The term chart is used interchangeably with the term Stateflow diagram and is used as a shorthand representation in the GUIs. The Stateflow block is a masked Simulink model and is equivalent to an empty, untitled Stateflow diagram. Use the Stateflow block to include a Stateflow diagram in a Simulink model. The term Stateflow diagram is used from the Stateflow point of view, whereas the term Stateflow block is used from the Simulink point of view.
How do I load a Stateflow diagram?	Stateflow diagrams exist only in Simulink models and are saved in the `.mdl` file along with the Simulink model information. When a Simulink model is loaded, any Stateflow diagrams contained within it are also loaded. To load a Stateflow diagram, you simply load its associated Simulink model.

Questions	Answers
How do I connect two states with a transition?	There are two ways to do this: Focus your mouse over an edge of the first state (the cursor should become a cross-hair indicating that a transition will be created) and click. You then drag your mouse until you hit an edge of the second state at which point the transition should snap into place. Alternatively, you can drag a default transition off the toolbar and snap it to the edge of a state. You will then need to connect the source of the default transition to the other state.
How do I copy data, events, and targets that I have created using the Explorer?	There is currently no mechanism for copying or pasting objects in the Explorer. To create a copy you need to explicitly create a duplicate object.
How do I change the font size of a collection of objects.	You can do this two ways: **1** Use the **Font Size** menu option on the graphics editor **Style** menu. **2** Use the graphics editor's context sensitive shortcut menu: **a** Right click in an open area of the graphics editor drawing area. A pop-up menu appears. **b** Click on the **Font Size** menu item and drag horizontally until the desired font size is displayed.

Questions	Answers
How do I change the decomposition of my states from exclusive (OR) to parallel (AND)?	To change the decomposition at a specified level in the hierarchy: 1 Select the parent state whose composition you want to change. 2 Display the shortcut menu by right clicking while the cursor is over the state. 3 Select **Parallel (AND)** from the shortcut menu.
How do I call Simulink from Stateflow and vice versa?	You can define data and events to be either **Input from Simulink** or **Output to Simulink**. You can use these objects to exchange information with the Simulink model.
How do I use MATLAB functions and variables in my Stateflow diagrams?	Stateflow provides two action language notations to use MATLAB functions and variables: • `ml()` functions • `ml.` name space operator See the *Stateflow User's Guide* for detailed information on these action language notations.
I defined several **Imported** and **Exported** events and data in a Stateflow diagram. Why can't I use them in Simulink blocks that belong to the same model?	Data and events parented by the machine can have **Exported** and **Imported** scope. You might think they are somehow shared with Simulink objects. It is important to note that **Imported**/**Exported** events and data have nothing to do with normal simulation. They are provided for use with stand-alone code generation for advanced users.

Questions	Answers
If I change a data **Workspace** object from the MATLAB command line during a simulation, will Stateflow see the change? If Stateflow changes a data **Workspace** object, will its value as accessed from the MATLAB command line change?	You need to define data **Workspace** objects in the MATLAB command window before starting the simulation. At the start of the simulation the value of the data **Workspace** object is evaluated and used as such by Stateflow. The Stateflow block may update the values of the data object; any changes can be observed at the MATLAB command line. However, if a data **Workspace** object is changed at the MATLAB command line during a simulation, it is ignored by Stateflow. That is, the Stateflow block continues to use the value as defined at the start of simulation or as changed by Stateflow during the simulation. After simulation starts, do not modify data **Workspace** objects from the MATLAB command line; this action will corrupt the link between the Stateflow block and the MATLAB workspace and any modified values computed by the Stateflow block will be lost.
What is the difference between the **Parse** and **Parse Diagram** menu items on the graphics editor **Tools** menu?	The **Parse** menu item parses all the Stateflow diagrams for the current model/machine. **Parse Diagram** parses the current diagram only.
I am having trouble generating code for the simulation target. What could be wrong?	Proper MEX setup is required for Stateflow to generate the S-function for simulation. Make sure that you have followed the recommended order of installation of MATLAB, Simulink, and Stateflow relative to the C compiler installation. (See the *MATLAB Application Program Interface Guide* for information about installing compilers.)

Questions	Answers
Some of the error messages related to code generation have what looks like ID numbers in them. What are those numbers?	Occasionally you will see numbers preceded by a # character in error messages. This number is a unique ID (or handle) that represents an object in the data dictionary. These IDs are unique to a particular session (similar to Handle Graphics object handles). The line in the pop-up window that contains the ID is a hypertext link. Double-clicking on that error line in the pop-up window brings the graphics editor window to the front, zooms on the relevant object, and selects the ID of the object causing the problem.
When I start simulation on my Stateflow diagram, I see the following error message in a dialog box. `“Error using => toolbox/stateflow/ stateflow/private/ (intfcn_method). Stateflow S-Function build failed."` Why?	Error messages related to code generation can originate from the parser, the simulation code generator, Stateflow Coder, or from the external C compiler. If you are explicitly parsing, generating code, or building a target, it is apparent which operation generated the error. It is less clear which operation generated the error if you are implicitly performing these operations through one of these methods: • Choosing **Start** from the graphics editor **Simulation** menu. • Choosing **Start** from the Simulink model **Simulation** menu. • Choosing **Debug** from the graphics editor **Tools** menu and clicking on the **Go** button. Errors are displayed (in red) in an informational dialog box and also in the MATLAB command window. Note that whenever you see a line in the error message dialog box, you can double-click on that line to zoom in on the object in the Stateflow diagram. See the *Stateflow User's Guide* for more information.

Questions	Answers
How do I animate my Stateflow diagram during simulation?	The `sfun` simulation target has debugging/animation enabled by default. However, if this setting has been changed, you may need to re-enable it. Invoke the Explorer and select the machine/model you want to simulate. 1 Choose **Open Simulation Target** from the **Tools** menu to display the **sfun Target Configuration properties** dialog box. 2 Enable animation and click on the **Apply** button. 3 Select **Debug** on the graphics editor **Tools** menu. 4 Ensure that animation is enabled in the Debugger window. Click on the **Go** button to run the simulation. If you encounter errors, see the *Stateflow User's Guide* for more information.

Questions	Answers
How do I incorporate my own code into the Stateflow generated code?	When you generate code for a Stateflow machine you use objects called *targets*. There are potentially several targets for any given machine. Stateflow provides a default simulation target called `sfun`. The option of creating multiple targets allows you to generate code in different ways. Essentially, target objects embody code generation configuration information. Each target object has an associated **Configuration properties** dialog box that includes a **Custom Code** field. You can incorporate your own code into the Stateflow generated code by entering the code in this field. The text placed in the **Custom Code** field is inserted at the top of the Stateflow generated source code. You may want to include some global variable declarations or preprocessor directives such as `#include` or `#define`. To include external custom libraries, use the `–l` MEX command option in the **Configuration properties** dialog box **Make Command** field. See the *Stateflow User's Guide* for information on displaying the various target **Configuration properties** dialog boxes.

Questions	Answers
For Simulink parameters I have a fixed-step solver with a 1 second step size. I drive the Stateflow block with a 1Hz pulse going into a rising edge trigger. Why doesn't the chart ever execute? Why doesn't it execute with a falling edge trigger either?	You've specified that the Stateflow block is to be triggered based on a 1Hz rising input event. The Simulink model solver and step size dictate how and when the Simulink model executes. When executed once per second, the 1Hz pulse looks like a constant to the rest of the system with no edges to trigger on. This combination of Simulink parameters and Stateflow block triggering does not allow the Stateflow block to execute. There is not enough time for the rising edge to be detected. Changing to falling edge doesn't work either for the same reason. The solver must execute at a rate greater than twice that of the highest system frequency of concern. In this case, the maximum step size must be less than 0.5 second.
How can I use a single **Output to Simulink** event to execute several subsystems in a predetermined order?	You can do this by cascading the subsystems through their triggers. Define the event as an **Output to Simulink** and include a trigger block within each of the subsystems. Connect the Stateflow event output directly to the trigger input of the first subsystem to be executed. Set up the subsystem trigger block to have **Show Output Port** selected and connect the output port as a subsystem output. Connect the subsystem output to the trigger input of the next subsystem to execute. Any number of subsystems can be chained together to control execution order. To use this method, the Simulink subsystem triggers must be specified as **function-call** or **either-edge**.

Questions	Answers
Pressing the **Esc** key seems to cause different behavior in Stateflow than it does in Simulink. What are the differences?	Pressing the **Esc** key in Simulink closes the current window and causes the parent of that window to come to the front. Repeatedly pressing the **Esc** key results in the same behavior until the root of the hierarchy is reached. Pressing the **Esc** key in Stateflow cancels the last operation and deselects any selected objects. Pressing the **Esc** key a second time causes the parent of that window (the Simulink model window) to come to the front. Stateflow does not close the graphics editor window. The primary difference is that Simulink closes lower level windows while proceeding up the hierarchy, while Stateflow does not.
Why can't I get a prompt in the MATLAB command window while I'm debugging?	When the Debugger is in operation, it is considered the current active figure. This means that input from the MATLAB command window is disabled. The Debugger provides a MATLAB command field in its main window to allow access to the MATLAB command line.
Where does the graph go when I plot from the Debugger MATLAB command line field?	Executing Handle Graphics calls while using Stateflow can give unexpected results because Stateflow GUIs are based on Handle Graphics. When you issue a plot command from the MATLAB command field in the Debugger, the Stateflow graphics editor window is considered the current figure. To have the graph appear in a different figure from the current one (the graphics editor window), explicitly open a new figure (e.g., `figure(2)`).

6

Toolboxes and Blocksets

Signal Processing Toolbox 4.1

SPTool Export Structures

The online (PDF) version of the *Signal Processing Toolbox User's Guide* now documents the SPTool export structures. See these sections in Chapter 5 of the PDF manual for complete information about the export structures:

- "Saving Signal Data" for information about the Signal export structure
- "Saving Filter Data" for information about the Filter export structure
- "Saving Spectrum Data" for information about the Spectrum export structure

DSP Blockset 2.2

Additional New Blocks

The DSP Blockset contains two new blocks that are not listed in *MATLAB 5.2 Product Family New Features*:

- Normalization
- Complex Normalization

Both blocks are in the Vector Math library. Please see the block reference for complete information about these blocks.